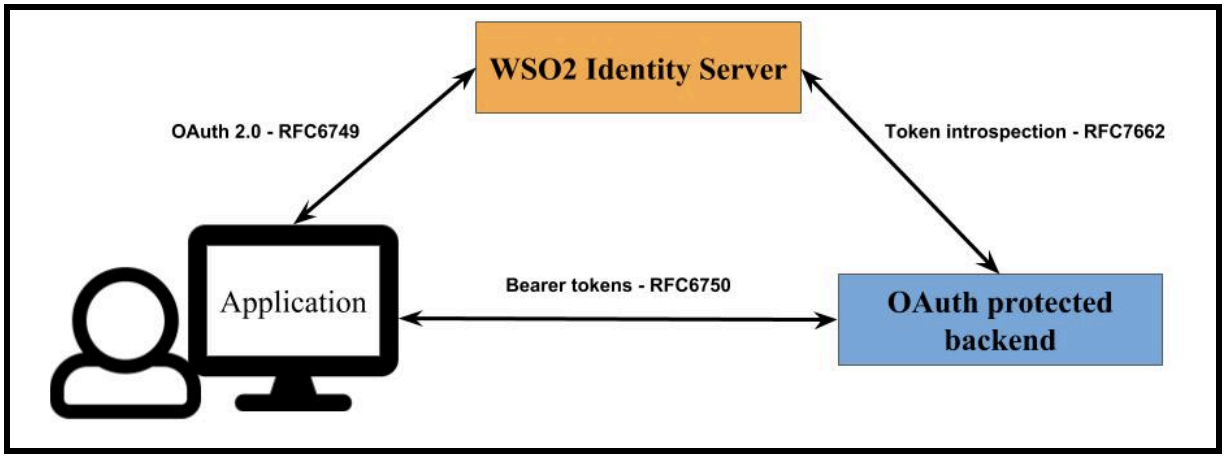
# Access Delegation - OAuth 2.0

## Introduction:

In this tutorial, you will experience how the WSO2 Identity Server (WSO2 IS) can be used as an OAuth 2.0 authorization server. This tutorial guides you through the OAuth 2.0 application configuration, deployment, and usage scenario.

In this sample, a user will try to login to a web application called pickup-dispatch via a WSO2 Identity Server and get an access token to call the API to view the vehicle bookings.



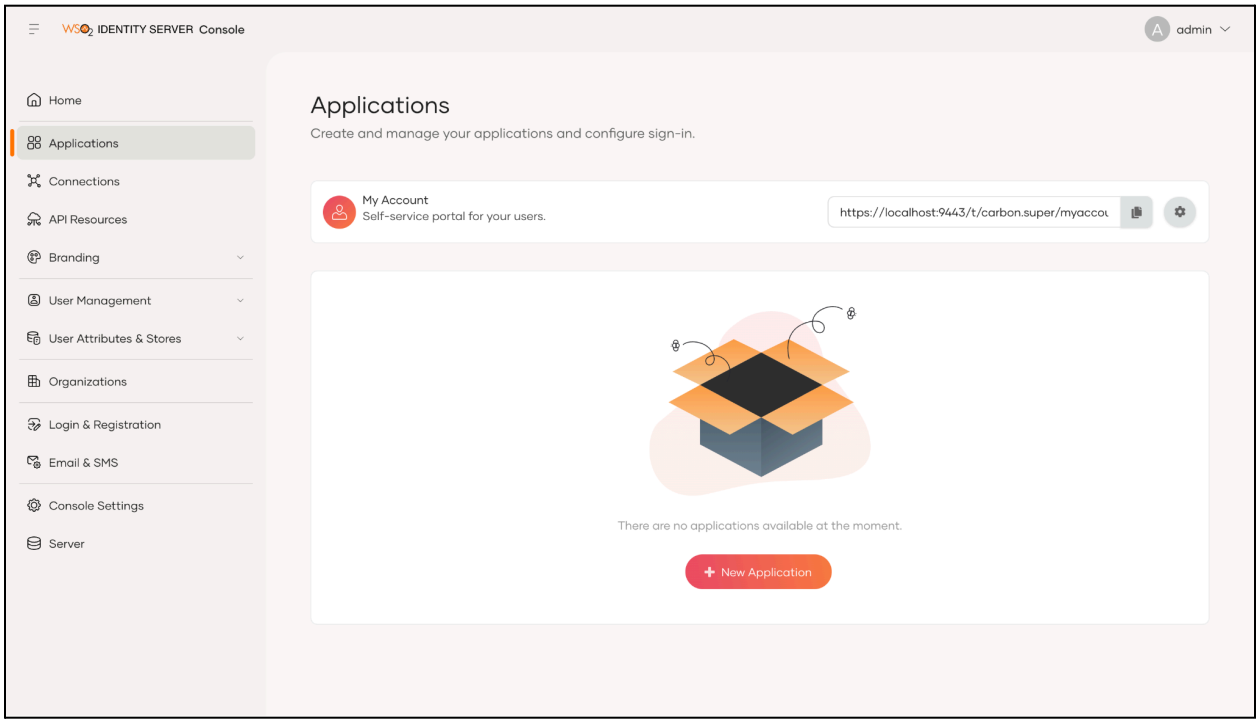
## Setting up:

1. Open the `/etc/hosts` file and add the following entry to map the new hostname.
    - If you're using Windows, the hosts file will be in the `c:\Windows\System32\Drivers\etc` directory.

```
127.0.0.1       localhost.com
```

2. Download the latest sample pickup-dispatch sample application from here.
3. Download a tomcat server 8 or 9, run the server on port 8080 and deploy the pickup-dispatch.war file.

   **ex**: Move the `pickup-dispatch.war` to the `<TOMACAT_PATH>/webapps`
4. Configure pickup dispatch application as a service provider in WSO2 Identity Server.
5. In the WSO2 Identity Server **Console,** from the menu click **Applications.**

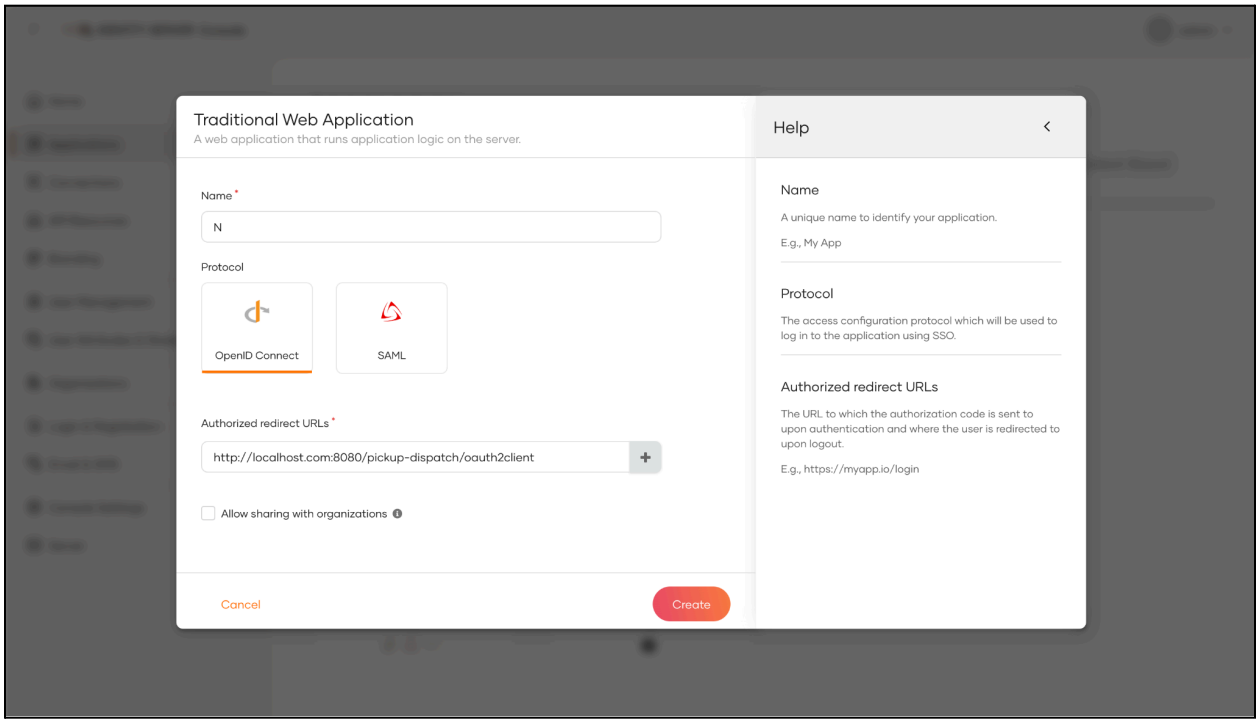



6. Click **New Application.**
7. From the given set of templates select **Traditional Web Application** template.
8. Fill the fields in the create application wizard.

    **Name**: *Pickup Dispatch*

    **Authorized redirect URLs:**

    *http://localhost.com:8080/pickup-dispatch/oauth2client*




9. Goto Protocol tab and note **Client ID** and **Client Secret**.
10. Extract **pickup-dispatch.war** and open **dispatch.properties** located at **<EXTRACT>/WEB-INF/classes**
11. Replace **consumerKey** and **consumerSecret** values with **Client ID** and **Client Secret** taken from the newly created service provider. And be sure the value of scope is “openid internal_application_mgt_view”.

```
consumerKey=7U1aEVaJroVYvXCCpoMFTs4uvO0a
consumerSecret=N4wMAVb9pkGXzjPuUm14UCJ_nH0qfGK0aaBv4aLntPsa

callBackUrl=http://localhost.com:8080/pickup-dispatch/oauth2client
scope=openid internal_application_mgt_view
authzGrantType=code

enableOIDCSessionManagement=false
enableOIDCBackchannelLogout=true
authzEndpoint=https://localhost:9443/oauth2/authorize
OIDC_LOGOUT_ENDPOINT=https://localhost:9443/oidc/logout
sessionIFrameEndpoint=https://localhost:9443/oidc/checksession
tokenEndpoint=https://localhost:9443/oauth2/token
claimManagementEndpoint=https://localhost:9443/services/ClaimMetadataManagementService

post_logout_redirect_uri=http://localhost.com:8080/pickup-dispatch/oauth2client
api_endpoint=http://localhost:39090/bookings

adminUsername=admin
adminPassword=admin
```

12. Restart the Apache Tomcat server.

    ex:- By executing **sh catalina.sh start** from bin folder

13. Download the **backend-service.jar** file from here.
14. Navigate to the location, where backend-service.jar resides and then start the backend service by executing the following command.

    (Tip - For more information about the backend service, see Introduction to Backend Service)

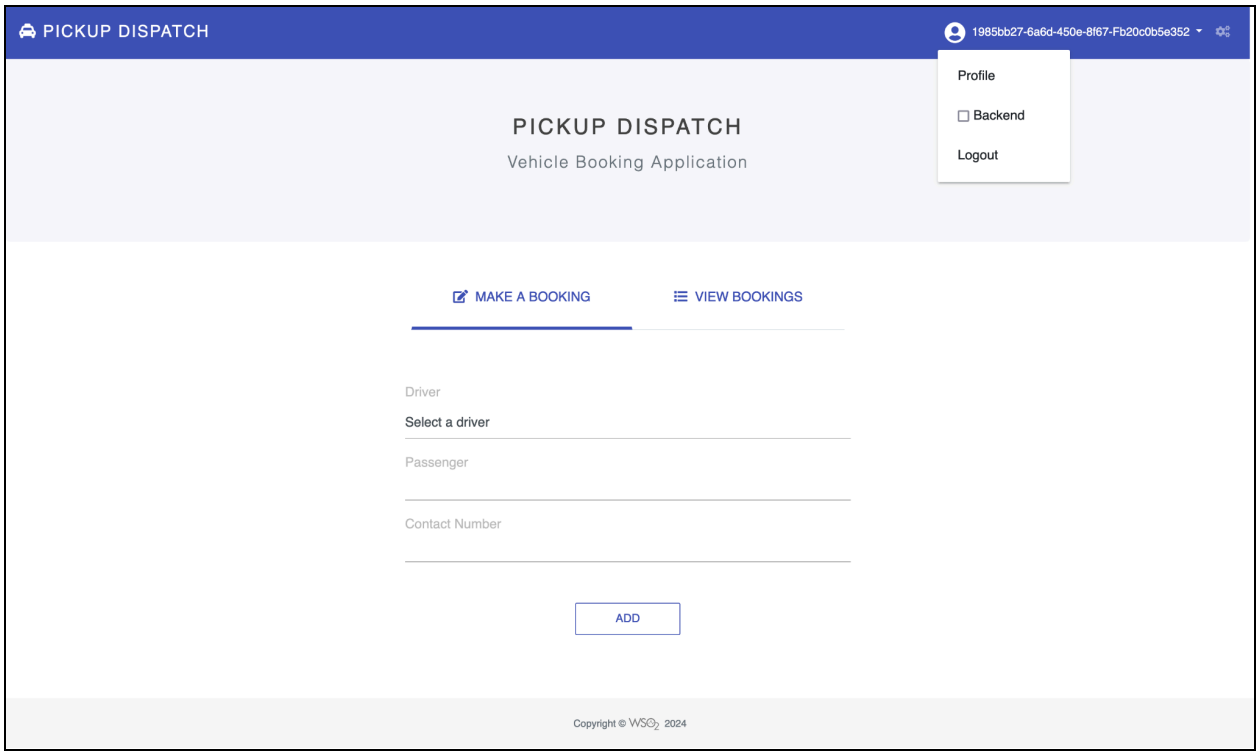
    **java -jar backend-service-<version>.jar**

    The setup is now complete and you can proceed to try out the scenario.

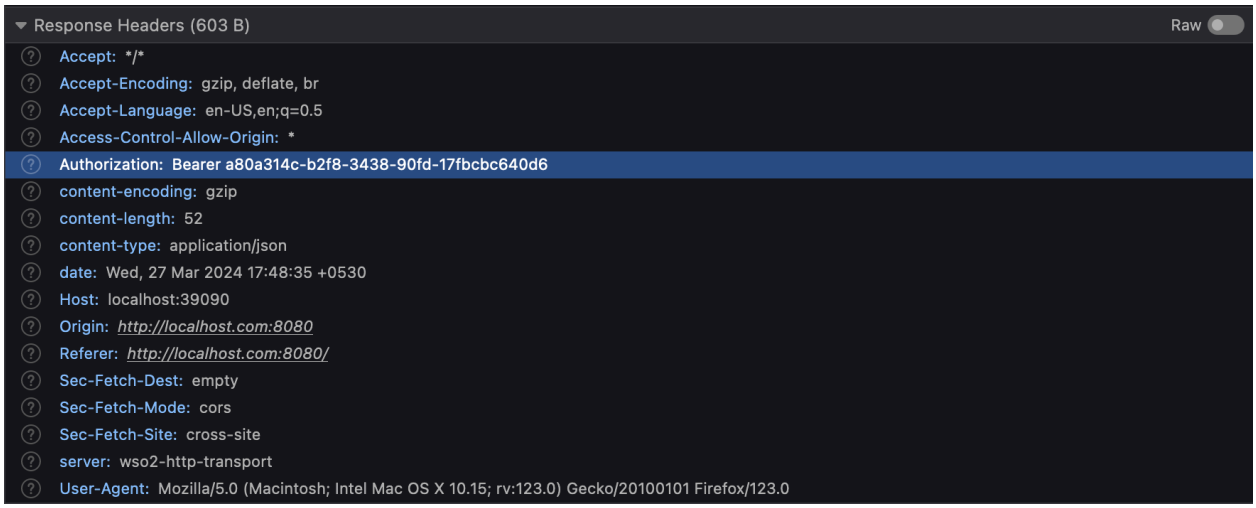
## Try It:

1. Run the application by visiting http://localhost.com:8080/pickup-dispatch/index.jsp URL and log in using preferred user credentials. (ex:- admin/admin)

2. By default, the backend calls are disabled. Therefore, enable it by ticking the **Backend** box in the drop-down list as shown in the diagram below:




3. Now we can add a new booking by selecting a driver from the drop-down list and filling passenger details. When you click on the Add button, you can notice the request sent to http://localhost.com:39090/bookings contains an authorization header.
4. Furthermore, the user should click on the gear icon on the top-right corner to view the request and response.
5. Similarly, you can view stored bookings. This request too contains an Authorization header.




If we try sending the same request without an authorization header you will get a 401 Unauthorized response.